Chip Placement with Deep Reinforcement Learning

Literature review:
3 popular methods: 1) partitioning-based methods, 2)stochastic/hill-climbing methods, 3) analytic solvers.
1) partitioning-based methods:
divide the huge net into pieces (like abstract the operational amplifier etc.) then use optimal solver
drawback: global routing congestion & local weakness influences the whole net

2) stochastic / hill-climbing algorithms (Simulated annealing 退火算法)

drawback: slow and difficult to parallelize
3) analytic solvers
electrostatics-based methods or predict the number of Design Rule Check (DRC) violations for a given macro placement.
This article proposes an end-to-end learning-based methods, closely related to analytic solvers

Methods

State: every possible partial placement of the netlist onto the chip canvas
Action: possible addition of macro
Reward: always 0 but the terminal state negative weighted sum of proxy wirelength and congestion (proxy cost)

How to train and learn?
Use 5 different accelerators to generate 10,000 chip placements with different congestion weights (ranging from 0 to 1) and random seeds, collecting these generation data as well. (value network outputs the generated net, policy network learns from the greedy value network)

Not clear
(nodes representation:
1) edge: fully connected network to an aggregated representation of intermediate node embeddings (e)
2) node: updates its representation by taking the **mean** of adjacent edge embeddings (v)

$$e_{ij} = fc_1(concat(fc_0(v_i)|fc_0(v_j)|w_{ij}^e)) \quad (5)$$
$$v_i = mean_{j \in \mathcal{N}(v_i)}(e_{ij})$$

)

Transfer learning to improve the transfer ability as well

Spatio-temporal evolutionary analysis of the township enterprises of Beijing suburbs using computational intelligence assisted design framework

computational intelligence aided design framework, which embedded the micro-genetic algorithm with variable populations

rapid development → chaotic development → steady development

Computational Intelligence Assisted Design
Char 22
PX denotes the long term rational price, BM denotes the equilibrium price after the game.

How to price:
If the generation is not used out (generating too much or less consumption), weighted average of accepted bids（买单报价）

If the generation is short (generating not enough or less consumption) weighted average of accepted offers (卖单报价)

opportunistic collusion（only increase product when making profits） & loyal co-operator (follow the agreement to add product)

trigger price strategies (Lower price will lead to punishment) Trigger price strategy depends on four parameters

Trigger price strategy depends on four parameters, P_{TR}, T, Q_{comp} and Q_{coop}, where P_{TR} is the trigger price, T is the number of time periods the punishment will last, Q_{comp} is the competitive output given 100 per cent generation volume $Q_{smax}{}^{i}$ and Q_{coop} is the cooperative output given $Q_{smax}{}^{i}$. $(1–X)$.

At PX, there are some stable generators so that long term price is low.

$$Q_{smax}{}^{i} \cdot l = Q_{SPX}{}^{i} + Q_{SBM}{}^{i}$$

portfolio instrument l expressed as a percentage of its total generation capacity (operation percentage)

$$C_L = \sum_{r}^{48} PXP^r . Q_D^r \quad (22.3)$$

where C_L is the marginal cost of supplier j, r is the settlement period number, PXP is the PX clearing price and Q_D^j is the actual demand at settlement period r:

$$C_S = \sum_{r}^{48} (PXP^r . Q_C^r - Max[0, Q_C^r - Q_D^r] . SSP^r + Max[0, Q_D^r - Q_C^r] . SBP^r) \quad (22.4)$$

where C_S is the contracted cost of supplier j, Q_C^j is the contracted volume at settlement period r on PX. A percentage premium for supplier's strategy is derived from (C_S / C_L); the lower the premium the more efficient the strategy.

genetic algorithm (GA) for opt